Gated RNN & Sequence Generation
Hung-yi Lee
李宏毅
Outline

• RNN with Gated Mechanism
• Sequence Generation
• Conditional Sequence Generation
• Tips for Generation
RNN with Gated Mechanism
Recurrent Neural Network

- Given function $f$: $h', y = f(h, x)$

$h$ and $h'$ are vectors with the same dimension

No matter how long the input/output sequence is, we only need one function $f$
Deep RNN

\[ h', y = f_1(h, x) \quad b', c = f_2(b, y) \]
Bidirectional RNN

\[ h', \ a = f_1(h, x) \]
\[ b', \ c = f_2(b, x) \]

\[ y = f_3(a, c) \]
Naïve RNN

- Given function \( f: h', y = f(h, x) \)

\[ h' = \sigma(W^h h + W^i x) \]

\[ y = \sigma(W^o h') \]

\text{softmax}

Ignore bias here
c changes slowly \( \Rightarrow \) \( c^t \) is \( c^{t-1} \) added by something

h changes faster \( \Rightarrow \) \( h^t \) and \( h^{t-1} \) can be very different
\[
z = \tanh(W x_t)
\]

\[
z^i = \sigma(W^i x_t)
\]

\[
z^f = \sigma(W^f x_t)
\]

\[
z^o = \sigma(W^o x_t)
\]
$z = \tanh(\begin{bmatrix} W \end{bmatrix} h^{t-1})$

diagonal

"peephole"

obtained by the same way
\[ c^t = z^f \odot c^{t-1} + z^i \odot z \]
\[ h^t = z^o \odot \text{tanh}(c^t) \]
\[ y^t = \sigma(W'h^t) \]
LSTM

\[ y^t \]

\[ h^t \]

\[ c^t \]

\[ x^t \]

\[h^{t+1} \]
GRU

\[ h^t = z \circ h^{t-1} + (1 - z) \circ h' \]
LSTM: A Search Space Odyssey
LSTM: A Search Space Odyssey

1. No Input Gate (NIG)
2. No Forget Gate (NFG)
3. No Output Gate (NOG)
4. No Input Activation Function (NIAF)
5. No Output Activation Function (NOAF)
6. No Peepholes (NP)
7. Coupled Input and Forget Gate (CIFG)
8. Full Gate Recurrence (FGR)

Standard LSTM works well
Simply LSTM: coupling input and forget gate, removing peephole
Forget gate is critical for performance
Output gate activation function is critical
An Empirical Exploration of Recurrent Network Architectures

<table>
<thead>
<tr>
<th>Arch.</th>
<th>Arith.</th>
<th>XML</th>
<th>PTB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tanh</td>
<td>0.29493</td>
<td>0.32050</td>
<td>0.08782</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.89228</td>
<td>0.42470</td>
<td>0.08912</td>
</tr>
<tr>
<td>LSTM-f</td>
<td>0.29292</td>
<td>0.23356</td>
<td>0.08808</td>
</tr>
<tr>
<td>LSTM-i</td>
<td>0.75109</td>
<td>0.41371</td>
<td>0.08662</td>
</tr>
<tr>
<td>LSTM-o</td>
<td>0.86747</td>
<td>0.42117</td>
<td>0.08933</td>
</tr>
<tr>
<td>LSTM-b</td>
<td>0.90163</td>
<td>0.44434</td>
<td>0.08952</td>
</tr>
<tr>
<td>GRU</td>
<td>0.89565</td>
<td>0.45963</td>
<td>0.09069</td>
</tr>
<tr>
<td>MUT1</td>
<td>0.92135</td>
<td>0.47483</td>
<td>0.08968</td>
</tr>
<tr>
<td>MUT2</td>
<td>0.89735</td>
<td>0.47324</td>
<td>0.09036</td>
</tr>
<tr>
<td>MUT3</td>
<td>0.90728</td>
<td>0.46478</td>
<td>0.09161</td>
</tr>
</tbody>
</table>

LSTM-f/i/o: removing forget/input/output gates
LSTM-b: large bias
Importance: forget > input > output
Large bias for forget gate is helpful
An Empirical Exploration of Recurrent Network Architectures

MUT1:

\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + b_z) \\
    r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + \tanh(x_t) + b_h) \odot z \\
    &\quad + h_t \odot (1 - z)
\end{align*}
\]

MUT2:

\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + W_{hz}h_t + b_z) \\
    r &= \text{sigm}(x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\
    &\quad + h_t \odot (1 - z)
\end{align*}
\]

MUT3:

\[
\begin{align*}
    z &= \text{sigm}(W_{xz}x_t + W_{hz} \tanh(h_t) + b_z) \\
    r &= \text{sigm}(W_{xr}x_t + W_{hr}h_t + b_r) \\
    h_{t+1} &= \tanh(W_{hh}(r \odot h_t) + W_{xh}x_t + b_h) \odot z \\
    &\quad + h_t \odot (1 - z)
\end{align*}
\]
Neural Architecture Search with Reinforcement Learning

LSTM

From Reinforcement Learning
Sequence Generation
Generation

- Sentences are composed of characters/words
- Generating a character/word at each time by RNN

\[
x: \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}
\]
\[
y: \begin{bmatrix} 0 & 0 & 0 & 0.7 & 0.3 & \cdots & 0 \end{bmatrix}
\]

Distribution over the token
(sampling from the distribution to generate a token)

The token generated at the last time step.
(represented by 1-of-N encoding)
Generation

- Sentences are composed of characters/words
- Generating a character/word at each time by RNN

\[ y^1: P(w | <BOS>) \]
\[ y^2: P(w | <BOS>, \text{床}) \]
\[ y^3: P(w | <BOS>, \text{床,前}) \]

Until \(<\text{EOS}>\) is generated

- sample, argmax
Generation

• Training

Training data: 春眠不覚曉

minimizing cross-entropy
Generation

- Images are composed of pixels
- Generating a pixel at each time by RNN

Consider as a sentence
blue red yellow gray ......
Train a RNN based on the “sentences”
Generation - PixelRNN

- Images are composed of pixels
Conditional Sequence Generation
Conditional Generation

• We don’t want to simply generate some random sentences.
• Generate sentences based on conditions:

**Caption Generation**

Given condition: A young girl is dancing.

**Chat-bot**

Given condition: “Hello”

“Hello. Nice to see you.”
Conditional Generation

- Represent the input condition as a vector, and consider the vector as the input of RNN generator

**Image Caption Generation**

Input image

A vector

A

woman

. (period)

<BOS>

......
Conditional Generation

• Represent the input condition as a vector, and consider the vector as the input of RNN generator
• E.g. Machine translation / Chat-bot
Conditional Generation

M: Hello
U: Hi
M: Hi

Need to consider longer context during chatting

https://www.youtube.com/watch?v=e2MpOmyQJw4

M: Hello
U: Hi

Dynamic Conditional Generation

Encoder

Decoder
Machine Translation

• Attention-based model

\[ \alpha = h^T W z \]

What is \( \text{match} \)?

- Jointly learned with other part of the network
- Cosine similarity of \( z \) and \( h \)
- Small NN whose input is \( z \) and \( h \), output a scalar

Design by yourself
Machine Translation

- **Attention-based model**

\[
c^0 = \sum \hat{\alpha}_0^i h^i = 0.5h^1 + 0.5h^2
\]
Machine Translation

- Attention-based model
Machine Translation

- Attention-based model

\[
c^1 = \sum \hat{\alpha}_1^i h^i = 0.5h^3 + 0.5h^4
\]
Machine Translation

• Attention-based model

The same process repeat until generating <EOS>
Speech Recognition

Image Caption Generation

A vector for each region

CNN

match → 0.7

$z^0$
Image Caption Generation

CNN

A vector for each region

Weighted sum

Word 1

\[ z^0 \rightarrow z^1 \]
Image Caption Generation

A vector for each region

\[ z^0 \rightarrow z^1 \rightarrow z^2 \]

Word 1 \( \rightarrow \) Word 2

weighted sum

0.0, 0.8, 0.2, 0.0

CNN

Filter

Filter

Filter

Filter

Filter

Filter

Filter

Filter
Image Caption Generation

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Image Caption Generation

A large white bird standing in a forest.
A woman holding a clock in her hand.
A man wearing a hat and a hat on a skateboard.
A person is standing on a beach with a surfboard.
A woman is sitting at a table with a large pizza.
A man is talking on his cell phone while another man watches.

Ref: A man and a woman ride a motorcycle
A man and a woman are talking on the road

Ref: A woman is frying food
Someone is frying a fish in a pot

Tips for Generation
Attention

**component**

\[ \alpha_t^i \]

**time**

**Bad Attention**

\[
\begin{align*}
\alpha_1^1 & \quad \alpha_2^1 & \quad \alpha_3^1 & \quad \alpha_4^1 \\
\alpha_1^2 & \quad \alpha_2^2 & \quad \alpha_3^2 & \quad \alpha_4^2 \\
\alpha_1^3 & \quad \alpha_2^3 & \quad \alpha_3^3 & \quad \alpha_4^3 \\
\alpha_1^4 & \quad \alpha_2^4 & \quad \alpha_3^4 & \quad \alpha_4^4
\end{align*}
\]

\[ w_1 \quad w_2 \, (\text{woman}) \quad w_3 \quad w_4 \, (\text{woman}) \quad \ldots \quad \text{no cooking} \]

**Good Attention**: each input component has approximately the same attention weight

E.g. Regularization term:

\[
\sum_i \left( \tau - \sum_t \alpha_t^i \right)^2
\]

For each component  
Over the generation

Mismatch between Train and Test

- **Training**

\[ C = \sum_t C_t \]

Minimizing cross-entropy of each component
Mismatch between Train and Test

**Generation**

We do not know the reference.

Testing: The inputs are the outputs of the last time step.

Training: The inputs are reference.

*Exposure Bias*
One step wrong

May be totally wrong

Never explore ……

一步錯，步步錯
Modifying Training Process?

When we try to decrease the loss for both steps 1 and 2 ..... 

Training is matched to testing. 

In practice, it is hard to train in this way.
Scheduled Sampling

From reference

From model

Exponential decay
Inverse sigmoid decay
Linear decay

0 200 400 600 800 1000

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Scheduled Sampling

- Caption generation on MSCOCO

<table>
<thead>
<tr>
<th></th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>CIDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always from reference</td>
<td>28.8</td>
<td>24.2</td>
<td>89.5</td>
</tr>
<tr>
<td>Always from model</td>
<td>11.2</td>
<td>15.7</td>
<td>49.7</td>
</tr>
<tr>
<td>Scheduled Sampling</td>
<td>30.6</td>
<td>24.3</td>
<td>92.1</td>
</tr>
</tbody>
</table>

Beam Search

The green path has higher score.
Not possible to check all the paths
Beam Search

Keep several best path at each step

Beam size = 2
Beam Search

The size of beam is 3 in this example.

https://github.com/tensorflow/tensorflow/issues/654#issuecomment-169009989
Better Idea?

I am ...... ✔ ✔
You are ...... ✔ ✔
You am ...... ✗ ✗

you

B

I ≈ you

A

A

B

A

B

high score

am ≈ are

you

A

B

A

B

you

B

B

l ≈ you

A

B

A

B

<BOSS>

A

B

A

B

<BOSS>

A

B

A

B
Object level v.s. Component level

- Minimizing the error defined on component level is not equivalent to improving the generated objects

Ref: The dog is running fast

\[ C = \sum_t C_t \]

Cross-entropy of each step

Optimize object-level criterion instead of component-level cross-entropy. Object-level criterion: \( R(y, \hat{y}) \)

\( y \): generated utterance, \( \hat{y} \): ground truth

Gradient Descent?
Reinforcement learning?

Start with observation $s_1$

Observation $s_2$

Observation $s_3$

Obtain reward $r_1 = 0$

Action $a_1$: "right"

Obtain reward $r_2 = 5$

Action $a_2$: "fire" (kill an alien)
Reinforcement learning?


The action we take influence the observation in the next step

reward: R(“BAA”, reference)
Concluding Remarks

- RNN with Gated Mechanism
- Sequence Generation
- Conditional Sequence Generation
- Tips for Generation